ELSEVIER

Artificial Intelligence in Engineering 14 (2000) 165–174

ARTIFICIAL INTELLIGENCE in ENGINEERING

www.elsevier.com/locate/aieng

Design of the speed controller for sensorless electric drives based on AI techniques: a comparative study

D. Kukolj[a], F. Kulic[a], E. Levi[b,*]

[a]*Faculty of Technical Sciences, University of Novi Sad, 21000 Novi Sad, Yugoslavia*
[b]*School of Engineering, Liverpool John Moores University, Liverpool L3 3AF, UK*

Received 18 November 1999; revised 10 May 2000; accepted 17 May 2000

Abstract

The paper investigates applicability of different artificial intelligence (AI) techniques in the design of a speed controller for electric drives. A speed-sensorless drive system is considered. A controller structure consisting of a load torque observer, a speed estimator and a speed predictor is developed. Next, different AI based approaches to speed controller design are investigated. The speed controllers based on (1) feed-forward neural network, (2) neuro-fuzzy network, and (3) self-organising Takagi–Sugeno (TS) rule based model are designed. A comparative analysis of the drive behaviour with these three types of AI based speed controllers is performed. In addition, a comparison is made with respect to the drive performance obtained with a conventional optimised PI controller. A detailed simulation study of a number of transients indicates that the best performance, in terms of accuracy and computational complexity, is offered by the self-organising Takagi–Sugeno controller. The controllers are developed and tested for a plant comprising a variable-speed separately excited DC motor.

Keywords: Feed-forward neural network; Fuzzy logic control; Neuro-fuzzy network; Takagi–Sugeno controller; Electric drives

1. Introduction

Conventional controllers, designed using the classical control theory, are well suited to control of linear processes whose exact model is known. However, the majority of physical systems usually contain non-linear relations that are difficult to model. Besides, in a number of cases, even when the available model is accurate enough, the exact system parameter values are difficult to obtain. Control structures based on artificial intelligence (AI), such as artificial neural networks (ANNs) [1] and fuzzy logic [2], appear to be an advantageous solution for control of such processes.

ANNs are suitable when the knowledge, contained within the available data sets, needs to be generalised. Their main drawback is that they act in a manner of a black box. Fuzzy logic deals with problems as a human mind, trying to handle vagueness, imprecision and uncertainty in a robust manner. However, the pure fuzzy logic systems have a disadvantage of being extremely difficult to tune for high performance. Fortunately, ANNs and fuzzy logic are complementary techniques and the shortcomings of one of them can be overcome by advantages of the other one. If ANNs are combined with fuzzy logic, neuro-fuzzy systems with learning capability are obtained [3].

ANNs and fuzzy logic are widely used in the areas of modelling, identification, diagnostics and control. Two classes of ANNs that have received considerable attention are the feed-forward neural networks (FNNs) and recurrent neural networks (RNNs). The FNNs are often used in identification [4–6], control [4–6], diagnostics [7] and prediction [8], although they perform static mapping. On the other hand, the RNNs, being dynamic networks, are of special interest in control [9,10]. Research on fuzzy logic applications in control shows a couple of general trends. The first and the oldest one is of heuristic nature. It is based on qualitative knowledge and experience of an expert with regard to the system behaviour, that is used in order to achieve a given control objective. ‘Trial and error’ approach is usually applied in this case in the analysis of the most appropriate domains for input/output fuzzy variables and in construction of the knowledge base [11,12]. A more general approach often utilises phase space [13] or response behaviour [14] as a mean of connecting the process dynamics and the rule base. The second, deterministic approach is based on a more or less systematic methodology

* Corresponding author. Tel.: +44-151-231-2257; fax: +44-151-298-2624.
E-mail address: e.levi@livjm.ac.uk (E. Levi).

0954-1810/00/$ - see front matter
PII: S0954-1810(00)00010-8

for identification of the structure and/or parameters of a fuzzy controller [15,16]. The third trend encompasses numerous methods for self-organising controllers and knowledge-base generation on the basis of given input–output numerical data [17,18]. Various optimisation methods, such as genetic algorithms [19] or simulated annealing [20], can be used during the course of tuning of the structure and/or parameters of a fuzzy controller. Finally, connection of fuzzy logic systems with ANNs yields neuro-fuzzy control structures of high performance [3,21].

This paper investigates the use of three different AI techniques in control of a plant comprising a variable-speed separately excited DC motor drive. All the three selected techniques are capable of acquiring and storing new knowledge about the process on the basis of representative numerical training samples. The first AI technique is the FNN, that is the most frequently applied type of ANNs. The second selected method is the adaptive neuro-fuzzy inference system (ANFIS) [3,22]. It represents an approach that combines ANNs and fuzzy logic, leading to an improvement over both of the individual methods. The third technique is a self-organising first-order Takagi–Sugeno (TS) fuzzy rule based model that relies on the product space clustering. The input–output space is partitioned using Gustafson–Kessel (GK) clustering algorithm [23]. This model is further combined with a FNN, with the aim of achieving an improvement in the performance.

A speed-sensorless control structure for a DC motor drive is initially designed, with the idea of performing a comparative study of the drive performance obtainable with speed control based on the listed three techniques. The control system structure includes a state observer, whose role is to provide the AI based speed controller with the information regarding the estimated load torque, estimated speed and predicted speed. The state observer is designed using three separate FNN structures. In order to provide a thorough and unbiased comparison, a conventional optimised PI controller is designed and investigated as well and its performance is included in the comparison.

Detailed consideration of a number of different operating regimes of the drive shows that, with respect to the behaviour in transients and steady states, self-organising TS and ANFIS speed controllers offer the best performance. It should be noted that the TS controller is especially advantageous since it is characterised with a quick training process, smaller computational requirements and a simple knowledge base.

The paper is organised as follows. Section 2 provides a short description of the three AI techniques considered in the design of the speed controller. The control structure of the DC drive, with three FNN based components of the state observer, is presented in Section 3. Design of the FNN, ANFIS and TS speed controllers is presented in Section 4. Simulation results, contained in Section 5, compare the performance of the DC drive obtained using the three AI based speed controllers and the optimised PI speed controller. Conclusions of the study are summarised in Section 6.

2. A review of AI techniques applied in the speed controller design

2.1. Artificial neural networks

Feed-forward ANNs [1] are used in this paper. The important features of the FNN, relevant for applications in electric motor drives, are its capability to perform non-linear mappings and its fault tolerance behaviour [4,5]. Although a FNN can not incorporate any dynamics within it, a non-linear dynamic system is emulated by time-delayed inputs. The training of the FNNs is performed using the well-known error back-propagation method [1,3].

2.2. Adaptive neuro-fuzzy inference system

An important advantage of neuro-fuzzy networks is that the heuristic knowledge regarding the process dynamics exists once when the training process is completed. Jang's ANFIS concept of a neuro-fuzzy network [3,22] is utilised here. This concept is based on the TS fuzzy reasoning in which the consequent of each rule is a linear combination of input variables plus a constant term, and the final output is a weighted average of each rule's output. The ANFIS is a multi-layer feed-forward adaptive network with five layers:

Layer 1. Every node in this layer contains a membership function (MF) which describes a grade of a linguistic label. The node functions are usually bell-shaped or triangular functions. The role of this layer is to perform, for a given number of inputs and given number of linguistic labels, a convenient partition of the multidimensional feature space. In contrast to the first layer, that represents the antecedent part of the fuzzy rule set, the remaining layers model the consequent part of the fuzzy rule set.
Layer 2. The firing strength of a rule is calculated in each node via the product operator (or, more generally, via *And* operator).
Layer 3. Every node of this layer calculates the normalised weight.
Layer 4. Each node in this layer has a node function of the first order. This function contains consequent parameters.
Layer 5. The single node in layer 5 computes the overall output as the sum of all the incoming signals from the preceding layer.

ANFIS training can be conducted in a number of ways. The hybrid learning algorithm [3] is utilised here. It combines the gradient descent and the least-square methods in order to find a feasible set of the antecedent and consequent parameters. The parameter values of the consequent part of fuzzy rules set are tuned during the forward pass of the hybrid learning algorithm, using a least-squares method

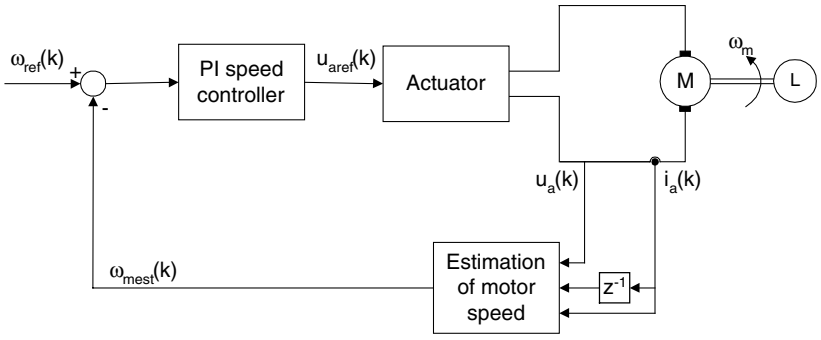

Fig. 1. Block diagram of the reference speed control system: drive with conventional PI speed controller and a speed estimator.

that minimises the mean squared error between the calculated and the desired output. Tuning of the parameters of the MFs from the first layer is performed in the backward pass, using a gradient descent method, such as error back-propagation.

2.3. Self-organising Takagi–Sugeno rule based model

The goal is to construct a TS fuzzy model of the first order that relies on the product space clustering [23]. This means that when K clusters are formed, the corresponding TS fuzzy model contains K rules of the following form:

$$\begin{aligned} &\mathrm{R}_i: \text{ If } x_1 \text{ is } A_{i1} \text{ and } x_2 \text{ is } A_{i2} \text{ and } \ldots \text{ and } x_n \text{ is } A_{in} \text{ then} \\ &y_i = \boldsymbol{x}\boldsymbol{a}_i + b_i, \quad i = 1, 2, \ldots K \end{aligned} \tag{1}$$

where R_i is the ith rule, $x_1, x_2, \ldots, x_n$ are input variables, $A_{i1}, A_{i2}, \ldots, A_{in}$ are the fuzzy sets assigned to the input variables, y_i is the value of the output of the ith rule, and $\boldsymbol{a}_i$ and b_i are parameters of the consequent function.

The procedure consists of two phases. The first phase identifies the structure of the rule base. During this phase, partition of the input–output space by means of GK clustering method takes place. The training data, that contain N input–output samples $\mathbf{z}_k = [x_k; y_k]^{\mathrm{T}}$, $k = 1 \ldots N$, are used. Each resulting cluster represents a certain operating region of the system, where the input–output data values are highly concentrated. These information clusters are interpreted as rules. The crucial part of the GK clustering algorithm [23] represents calculation of the distance from the sample under consideration to the centre of a cluster. This distance is calculated in the given lth iteration of the GK algorithm using:

$$\begin{aligned} &d_{ik}^2 = [\mathbf{z}_k - \mathbf{v}_i^{(1)}]^{\mathrm{T}} \{[\det(\mathbf{F}_i)^{(1/(n+1))}][(\mathbf{F}_i)^{-1}]\}[\mathbf{z}_k - \mathbf{v}_i^{(1)}], \\ &i = 1, 2, \ldots, K; \quad k = 1, 2, \ldots N, \end{aligned} \tag{2}$$

where $\boldsymbol{F}_i$ is covariance matrix for the ith cluster, and v_i is the centre of the ith cluster. Completion of the GK clustering process yields fuzzy partition matrix $\boldsymbol{U}$. This matrix contains values of the levels of belonging of all the samples to the appropriate clusters.

Once when the structure of the fuzzy model is defined, the second phase, parameter identification, is initiated. This phase enables calculation of the parameters that are present in the antecedent and consequent parts of the TS fuzzy rules. One-dimensional fuzzy sets in antecedent part of the ith rule A_{ij} are obtained from multidimensional fuzzy clusters, on the basis of matrix $\boldsymbol{U}$, by point-wise projection onto the space of the input variable x_j. In order to determine parameters of the consequent function, it is necessary to calculate normalised firing strength β_i of each rule for all the input samples. By forming the matrix composition of normalised firing strengths, it is possible to determine the consequent parameters using least-squares method [3].

The functional dependence between the input and output of the system is established once when the parameters $\boldsymbol{a}_i$ and b_i of each ith rule are calculated. It is possible now to calculate the aggregated output of the model as the weighted average of the rule consequents:

$$y_k = \frac{\sum_{i=1}^{K} [\beta_i(\mathbf{x}_k) \cdot (\mathbf{x}_k \boldsymbol{a}_i + b_i)]}{\sum_{i=1}^{K} \beta_i(\mathbf{x}_k)} \tag{3}$$

3. Description of the drive control structures

Two control structures of a separately excited DC motor drive are discussed. The first one is regarded as the reference control structure and it is introduced for the sake of comparative analysis. The reference control structure, shown in Fig. 1, utilises the conventional PI speed controller (drive $\mathrm{D}_{\mathrm{PI\text{-}C}}$). The PI speed controller is designed using the symmetrical optimum criterion [24]. An appropriate anti wind-up mechanism is used [24]. The following constraint, governed by the physical properties of the system, is set in simulations: maximum armature voltage is limited to 120% of the rated armature voltage ($u_{\mathrm{amax}} = 1.2 U_{\mathrm{an}}$). The block 'Actuator' in Fig. 1 denotes the power supply of the motor. Motor speed of rotation is estimated from the sampled armature voltage and current,

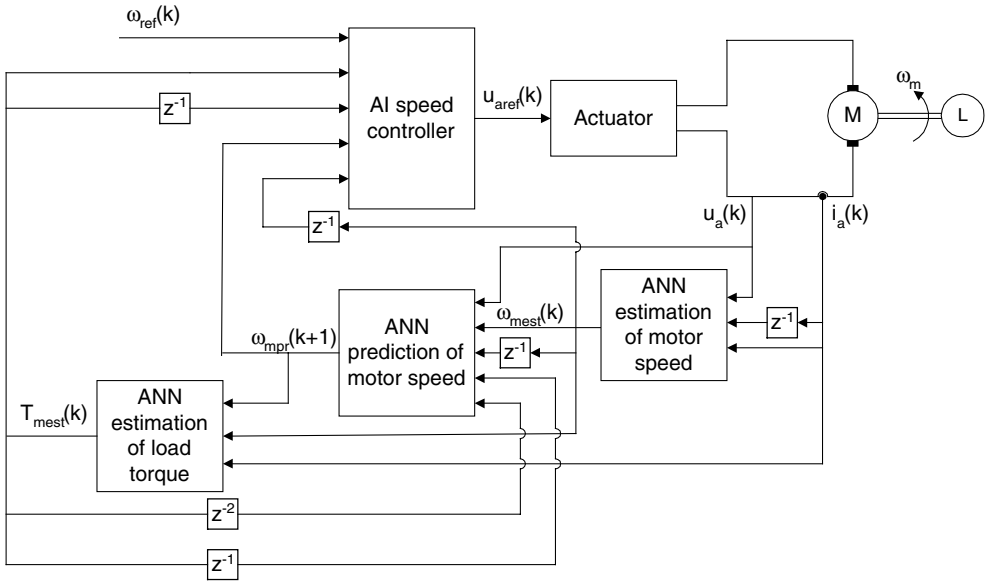

Fig. 2. Block diagram of the AI based control structures for the DC motor drive (the difference $\Delta\omega = \omega_{mpr}(k) - \omega_{mest}(k-1)$ serves as an input in the $D_{TS\text{-}C}$ drive).

as shown in Fig. 1, so that a sensorless drive is under consideration.

The second drive control structure, depicted in Fig. 2, contains a state observer and the AI based speed controller. Three speed controllers are designed: FNN (drive $D_{NN\text{-}C}$), ANFIS (drive $D_{NF\text{-}C}$), and the first-order TS rule based model (drive $D_{TS\text{-}C}$). The first two types of the speed controllers are characterised with five inputs each. These are the reference speed in the kth instant ($\omega_{ref}(k)$), the predicted speed in the kth instant ($\omega_{mpr}(k)$), the estimated speed in the $k-1$th instant ($\omega_{mest}(k-1)$) and the estimates of the load torque in the instants k and $k-1$ ($T_{mest}(k)$, $T_{mest}(k-1)$). The speed controller of the drive $D_{TS\text{-}C}$ has one input less, since the difference of speeds $\omega_{mpr}(k) - \omega_{mest}(k-1)$ serves as the input, rather than the individual values of these two signals. Each speed controller has a single output that represents the value of the control signal in the kth instant ($u_{aref}(k)$).

The control structure includes a state observer. The state observer contains three components [25]: a speed estimator, a speed predictor, and a load torque estimator. The speed predictor is essentially the model of the control object, whose role is to predict the motor speed. The torque estimator estimates the load torque in the transient and steady state operation. The load torque estimator is designed on the basis of the mechanical motion equation of the motor model. The outputs of these two components, together with the output of the speed estimator, are the inputs of the speed controller.

The state observer components are realised by means of FNNs that were trained using error back-propagation method [1,3]. The data used in the training and testing of the neural networks were obtained on random, using the discretised DC motor model and accounting for the stated limits. The training set consisted of 800 samples, while the test set contained 100 samples. The training and testing sets are different, but the values of individual variables belong to the same intervals (all variables are normalised and belong to the set $[-1,1]$).

In what follows, the results of the simulations conducted in order to select the most appropriate structures of the FNN for different components of the state observer are given. Apart from the accuracy, the selection criteria include simplicity of the structure, type of the activation functions of the neurons and the duration of the training process. As the best results were achieved with FNN structures with purely linear activation functions in the hidden and the output layer, the tables that follow contain only such configurations. The following notation is used:

- Structure of the ANN: $x_0 - x_1 - (x_2) - y$, where x_0 is the number of neurons in the input layer; x_1, (x_2) the number of neurons in the hidden layer(s); y the number of neurons in the output layer.
- Train.MSE—Mean square error per sample at the end of training.
- Test.MSE—Mean square error per sample in the testing.

3.1. Selection of the FNN for speed estimation

The FNN for the speed estimation is characterised with three inputs and one output (Fig. 2). The inputs are the armature current in the two consecutive instants ($i_a(k)$, $i_a(k-1)$) and the armature voltage ($u_a(k)$). The output is the estimated speed $\omega_{mest}(k)$. Characteristics of the training process and the testing are given in Table 1. The structure, selected for the speed estimator, is the one with three neurons in the hidden layer (the bold row in the Table 1).

Table 1
Data regarding training and testing of the FNNs for the speed estimation

Network structure	No. epochs	Train.MSE	Test.MSE
3–3–1	**4622**	$\mathbf{1.2497\times10^{-8}}$	$\mathbf{1.1323\times10^{-8}}$
3–5–1	4117	1.2484×10^{-8}	1.1312×10^{-8}
3–7–1	3592	1.2489×10^{-8}	1.1315×10^{-8}
3–9–1	3233	1.2485×10^{-8}	1.1315×10^{-8}
3–5–2–1	11 969	1.2496×10^{-8}	1.1423×10^{-8}
3–5–3–1	6421	1.2880×10^{-8}	1.1534×10^{-8}
3–7–2–1	3915	1.2494×10^{-8}	1.1346×10^{-8}
3–7–3–1	6122	1.2497×10^{-8}	1.1367×10^{-8}

3.2. Selection of the FNN for the speed prediction

The FNN for the speed prediction has five inputs and one output (Fig. 2). The inputs are the estimated speed in the two consecutive instants $\omega_{mest}(k)$, $\omega_{mest}(k-1)$, the estimated load torque in the two previous $k-1$ and $k-2$ consecutive instants ($T_{mest}(k-1)$, $T_{mest}(k-2)$) and the armature voltage in the kth instant ($u_a(k)$). The output is the predicted speed in the subsequent instant $k+1$ ($\omega_{mpr}(k+1)$). The results of the training and testing are summarised in Table 2. The FNN with seven neurons in the hidden layer is selected for the speed predictor (the bold row in Table 2).

3.3. Selection of the FNN for the torque estimation

The FNN for the torque estimation is characterised with three inputs and one output (Fig. 2). The inputs are the predicted speed in instant $k+1$ ($\omega_{mpr}(k+1)$), the estimated speed in the kth instant ($\omega_{mest}(k)$) and the armature current in the kth instant ($i_a(k)$). The output is the torque estimate in the kth instant ($T_{mest}(k)$). The results of the training and testing are given in Table 3. The FNN with two neurons in the hidden layer (bold row in Table 3) is selected as the FNN based torque estimator.

4. Selection of the structure of the AI based speed controllers

4.1. Feedforward neural network

As already noted the FNN for the speed control has five inputs and one output. The inputs are $\omega_{ref}(k)$, $\omega_{mest}(k-1)$, $\omega_{mpr}(k)$, $T_{mest}(k)$, and $T_{mest}(k-1)$. The output is $u_{aref}(k)$.

Table 2
Data regarding training and testing of the FNNs for the speed prediction

Network structure	No. epochs	Train.MSE	Test.MSE
5–2–1	100 000	9.4176×10^{-5}	9.7908×10^{-5}
5–3–1	100 000	8.8479×10^{-5}	9.1954×10^{-5}
5–5–1	100 000	1.0937×10^{-6}	1.1044×10^{-6}
5–7–1	**100 000**	$\mathbf{1.9912\times10^{-7}}$	$\mathbf{2.2165\times10^{-7}}$
5–9–1	100 000	1.2515×10^{-6}	1.3919×10^{-6}
5–7–2–1	100 000	1.3167×10^{-5}	1.3836×10^{-5}
5–7–3–1	100 000	8.0370×10^{-4}	8.1071×10^{-4}

Table 3
Data regarding training and testing of the FNN for torque estimation

Network structure	No. epochs	Train.MSE	Test.MSE
3–2–1	**4125**	$\mathbf{1.2438\times10^{-8}}$	$\mathbf{1.1520\times10^{-8}}$
3–3–1	4573	1.2454×10^{-8}	1.1414×10^{-8}
3–5–1	3829	1.2373×10^{-8}	1.1408×10^{-8}
3–7–1	3574	1.2492×10^{-8}	1.1511×10^{-8}
3–9–1	3131	1.2417×10^{-8}	1.1398×10^{-8}
3–5–2–1	12 199	1.2495×10^{-8}	1.1532×10^{-8}
3–5–3–1	4066	1.2388×10^{-8}	1.1524×10^{-8}
3–7–2–1	3813	1.2350×10^{-8}	1.1410×10^{-8}
3–7–3–1	5170	1.2318×10^{-8}	1.1401×10^{-8}

The data for training and testing were obtained, as noted, on random, by simulating the discrete DC motor model with the following constraints: $\omega_{ref}(k)\in[-1,1]$; $\omega_{mest}(k)\in[-1,1]$; $\omega_{mpr}(k)\in[-1,1]$; $T_{mest}(k)\in[-1,1]$; $T_{mest}(k-1)\in[-1,1]$; $\Delta T_{mestmax}\leq 0.1$. The results of the training and testing are given in Table 4. On the basis of the results of Table 4, the selected FNN speed controller structure is the one with three neurons in the hidden layer (bold row in Table 4). Activation functions in the hidden and output layer are linear.

4.2. Adaptive neuro-fuzzy speed controller

The inputs of the ANFIS based speed controller are the same as in the case of the FNN speed controller. The same data as in Section 4.1 are used for the training and testing of the ANFIS speed controller and the results are summarised in Table 5.

It is possible to determine the number of MFs and rules during design of the ANFIS controller in two ways. In the first case the number and the type of MFs are given in advance. Therefore it follows from the structure of the ANFIS network that the number of rules is given with p^q, where p is the number of MFs and q is the number of inputs of the ANFIS. The shortcoming of this approach is that the large number of inputs (five here) leads to a large ANFIS network, so that the training process is long and there is only a minute improvement in the accuracy. The problem can be overcome if a clustering algorithm (for example, subtructive clustering [3]) is used for the analysis of the training set. The number of clusters determines the number of rules, that is, the number and the type of the MFs. It was established during the course of this study that the ANFIS determined by means of a clustering algorithm yields the best performance if there are 13 clusters—rules and if the MFs are bell-like (the last row of Table 5).

4.3. The first order Takagi–Sugeno rule based model

The speed controller based on TS rule based model has four inputs and one output, as already noted and explained. The same training and testing sets as before are used again. Table 6 summarises the results of the training for different numbers of clusters. It follows from Table 6 that an increase

Table 4
Data regarding training and testing of the FNNs for the speed controller

FNN structure	Activ. function in layers		No. epochs	Train.MSE	Test.MSE
	Hidden	Output			
5–3–1	Logistic	Linear	100 000	2.0625×10^{-5}	2.0250×10^{-5}
5–3–1	**Linear**	**Linear**	**37 414**	$\mathbf{1.2500 \times 10^{-8}}$	$\mathbf{1.1505 \times 10^{-8}}$
5–3–1	tg hyperbolic	Linear	100 000	4.1406×10^{-7}	3.0967×10^{-7}
5–5–1	Logistic	Linear	100 000	3.2500×10^{-6}	4.0861×10^{-6}
5–5–1	Linear	Linear	26 379	1.2500×10^{-8}	1.1439×10^{-8}
5–5–1	tg hyperbolic	Linear	100 000	2.7209×10^{-7}	2.6652×10^{-7}
5–7–1	Logistic	Linear	100 000	6.1250×10^{-6}	5.4830×10^{-6}
5–7–1	Linear	Linear	15 387	1.2500×10^{-8}	1.1519×10^{-8}
5–7–1	tg hyperbolic	Linear	100 000	2.9954×10^{-7}	2.4033×10^{-7}
5–9–1	Logistic	Linear	100 000	7.2500×10^{-6}	6.7121×10^{-6}
5–9–1	Linear	Linear	35 078	1.2499×10^{-8}	1.1457×10^{-8}
5–9–1	tg hyperbolic	Linear	100 000	4.1050×10^{-7}	3.9137×10^{-7}
5–11–1	Logistic	Linear	100 000	1.7125×10^{-5}	2.2111×10^{-5}
5–11–1	Linear	Linear	25 808	1.2500×10^{-8}	1.1425×10^{-8}
5–11–1	tg hyperbolic	Linear	100 000	1.1584×10^{-6}	1.2064×10^{-6}

in the number of clusters does not lead to an increase in the accuracy of the training data approximation.

The TS rule based model with two clusters is adopted, so that there are two rules and two MFs per input. The consequent parts of the both rules differ insignificantly, so that the following expression can be adopted:

$$u_{\mathrm{aref}}(k) = 0.9518\omega_{\mathrm{ref}}(k) + 0.1524\Delta\omega(k) + 0.0593T_{\mathrm{mest}}(k) - 0.0059T_{\mathrm{mest}}(k-1) - 2.27e \times 10^{-10}. \quad (4)$$

A relatively simple analytical expression for the control of the plant is obtained in this way.

The antecedent part of the rule base is contained in the partition matrix $\boldsymbol{U}$. That is the most pronounced shortcoming of the described procedure for the TS fuzzy model generation (Section 2.3), related to the implementation. Number of samples and number of clusters determine dimensions of the partition matrix. A large memory space is therefore required for a large number of samples. This shortcoming is eliminated by using an FNN as the approximator of the MF of all the samples to the given fuzzy cluster. An FNN, with one hidden layer and the number of outputs equal to the number of clusters in the model, is formed in this way. Utilisation of the memory space is therefore greatly improved, without any adverse effect on the accuracy of the TS model operation.

5. Results of the simulation study

The standard second-order motor model, for operation with constant excitation flux, is utilised [26]. Mathematical model and all the relevant data of the motor are given in the Appendix A. All the described speed controller structures are elaborated in simulation. Twelve different operating regimes are considered:

1. Response to the rated step speed command ($\omega_{\mathrm{ref}}(t) =$ 1[r.u.]), with the constant rated load torque ($T_{\mathrm{m}}(t) =$ 1[r.u.]).
2. Response to the rated step speed command ($\omega_{\mathrm{ref}}(t) =$ 1[r.u.]) under no-load conditions.
3. Response to the rated step speed command ($\omega_{\mathrm{ref}}(t) =$ 1[r.u.]) under no-load conditions, followed by the step loading (rated load torque) and subsequent step unloading of the motor.
4. Response to the small step speed command ($\omega_{\mathrm{ref}}(t) =$ 0.1[r.u.]) with the constant rated load torque ($T_{\mathrm{m}}(t) =$ 1[r.u.]).
5. Response to the small step speed command ($\omega_{\mathrm{ref}}(t) =$ 0.1[r.u.]), $T_{\mathrm{m}}(t) = 0$[r.u.]).
6. Response to the small step speed command ($\omega_{\mathrm{ref}}(t) =$ 0.1[r.u.]), followed by the step application of the rated load torque and subsequent removal of the load.
7. Response to the rated speed command ($\omega_{\mathrm{ref}}(t) =$

Table 5
Data regarding training and testing of the ANFIS speed controller

MF shape	MF per entry	Rules	Epochs	Train.MSE	Test.MSE
Trapezoidal	2	32	10	2.0114×10^{-11}	1.7285×10^{-11}
Triangular	2	32	10	3.4942×10^{-12}	3.3568×10^{-12}
Bell-shaped	2	32	10	7.9282×10^{-10}	9.6323×10^{-10}
Bell-shaped	**13**	**13**	**10**	$\mathbf{5.1001 \times 10^{-13}}$	$\mathbf{5.9740 \times 10^{-13}}$

Table 6
Training error as function of the number of clusters

No. clusters	Max. MSE	MSE
2	$\mathbf{8.9244 \times 10^{-9}}$	$\mathbf{1.2175 \times 10^{-10}}$
3	8.7383×10^{-9}	1.2078×10^{-10}
4	8.7176×10^{-9}	1.2022×10^{-10}
5	9.0548×10^{-9}	1.1995×10^{-10}
6	8.8273×10^{-9}	1.1952×10^{-10}

1[r.u.]), followed by the speed reference halving ($\omega_{\text{ref}}(t) = 0.5$[r.u.]), and subsequent return of the speed reference to the rated value ($\omega_{\text{ref}}(t) = 1$[r.u.]). The load torque is constant and equal to the rated value at all times.

8. Operation of the drive with a time varying speed reference $\omega_{\text{ref}}(t) = 0.75 + 0.10 \sin(2\pi t/4) + 0.16 \sin(2\pi t/7)$[r.u.], with the rated load torque ($T_{\text{m}}(t) = 1$[r.u.]).
9. Response to the rated speed command $\omega_{\text{ref}}(t) = 1$[r.u.] and subsequent steady state operation with a time varying load torque $T_{\text{m}}(t) = 0.7 + 0.3 \sin(0.375t)$[r.u.].
10. Start-up of the motor with the small reference speed and operation with a time varying load torque, $\omega_{\text{ref}}(t) = 0.1$[r.u.]; $T_{\text{m}}(t) = 0.7 + 0.3 \sin(0.375t)$[r.u.].
11. Operation of the drive with the time varying speed reference and a speed dependent load torque: $\omega_{\text{ref}}(t) = 0.75 + 0.10 \sin(2\pi t/4) + 0.16 \sin(2\pi t/7)$[r.u.], $T_{\text{m}}(t) = 0.7 \text{ sign}(\omega_{\text{m}}(t))\omega_{\text{m}}^2(t)$[r.u.].
12. Operation of the drive with the time varying speed reference and a time varying, speed independent load torque: $\omega_{\text{ref}}(t) = 0.75 + 0.10 \sin(2\pi t/4) + 0.16 \sin(2\pi t/7)$ [$r.u.$], $T_{\text{m}}(t) = 0.7 + 0.3 \sin(0.375t)$[r.u.].

All the described drive structures are simulated for all the twelve listed operating regimes. For the sake of better quality of the comparison, the conventional PI speed controller of the $D_{\text{PI-C}}$ drive is optimised individually for each of the 12 operating conditions. Optimisation is performed using static Nelder–Mead (Simplex) method [27]. The performance index, used in the optimisation, is given with

$$I = \sum_{i=1}^{T} e_i^2 + 20 \cdot \sum_{i=T}^{\infty} e_i^2, \tag{5}$$

where T is the time instant determined with the crossover of the reference speed and actual speed after the first overshoot and e_i is the difference between the reference and the actual speed. The selected performance index has proved to be a good measure of the reduction of the transient process duration and of the overshoot for the most of the operating regimes under consideration.

Performance index (5) is calculated for all the operating regimes and for all the drive configurations (i.e. in each simulation run). Absolute and relative values of the performance index are given in Table 7 for the 12 considered regimes and for all the four types of the speed controller. The type of the speed controller with the minimum value of

Table 7
Performance index values for the considered operating regimes, obtained with the four different speed controllers

Operating regime	Performance indices			
	$D_{\text{PI-C}}$	$D_{\text{NN-C}}$	$D_{\text{NF-C}}$	$D_{\text{TS-C}}$
1	6.0833×10 (100%)	6.1060×10 (100.37%)	6.0844×10 (100.02%)	$\mathbf{6.0832 \times 10}$ **(99.99%)**
2	$\mathbf{3.6823 \times 10}$ **(100%)**	4.3702×10 (118.68%)	3.6824×10 (100.01%)	3.6840×10 (100.05%)
3	5.0788×10 (100%)	5.2593×10 (103.55%)	$\mathbf{4.2840 \times 10}$ **(84.35%)**	4.4679×10 (87.97%)
4	7.4378×10^{-2} (100%)	7.6210×10^{-1}. (1024.63%)	5.3040×10^{-2} (71.31%)	$\mathbf{3.4381 \times 10^{-2}}$ **(46.22%)**
5	$\mathbf{2.0029 \times 10^{-2}}$ **(100%)**	3.5405×10^{-1} (1767.69%)	2.5463×10^{-2} (127.13%)	2.4231×10^{-2} (120.98%)
6	1.4125×10 (100%)	1.2436×10 (88.04%)	$\mathbf{5.4048 \times 10^{-1}}$ **(38.26%)**	8.0847×10^{-1} (57.24%)
7	7.1265×10 (100%)	1.9320×10^{1} (271.10%)	6.7075×10 (94.12%)	$\mathbf{6.5904 \times 10}$ **(92.48%)**
8	7.8228×10^{1} (100%)	4.4756×10^{1} (57.21%)	4.5335×10^{1} (57.95%)	$\mathbf{4.4540 \times 10^{1}}$ **(56.94%)**
9	5.0226×10 (100%)	5.1349×10 (102.24%)	$\mathbf{5.0216 \times 10}$ **(99.98%)**	5.0217×10 (99.98%)
10	6.4027×10^{-2} (100%)	6.2816×10^{-1} (981.09%)	4.7221×10^{-2} (73.75%)	$\mathbf{4.1686 \times 10^{-2}}$ **(65.11%)**
11	3.0552×10^{1} (100%)	1.8880×10^{1} (61.80%)	1.8125×10^{1} (59.32%)	$\mathbf{1.6445 \times 10^{1}}$ **(53.82%)**
12	5.4999×10^{1} (100%)	3.1382×10^{1} (57.06%)	3.1717×10^{1} (57.67%)	$\mathbf{3.0954 \times 10^{1}}$ **(56.28%)**

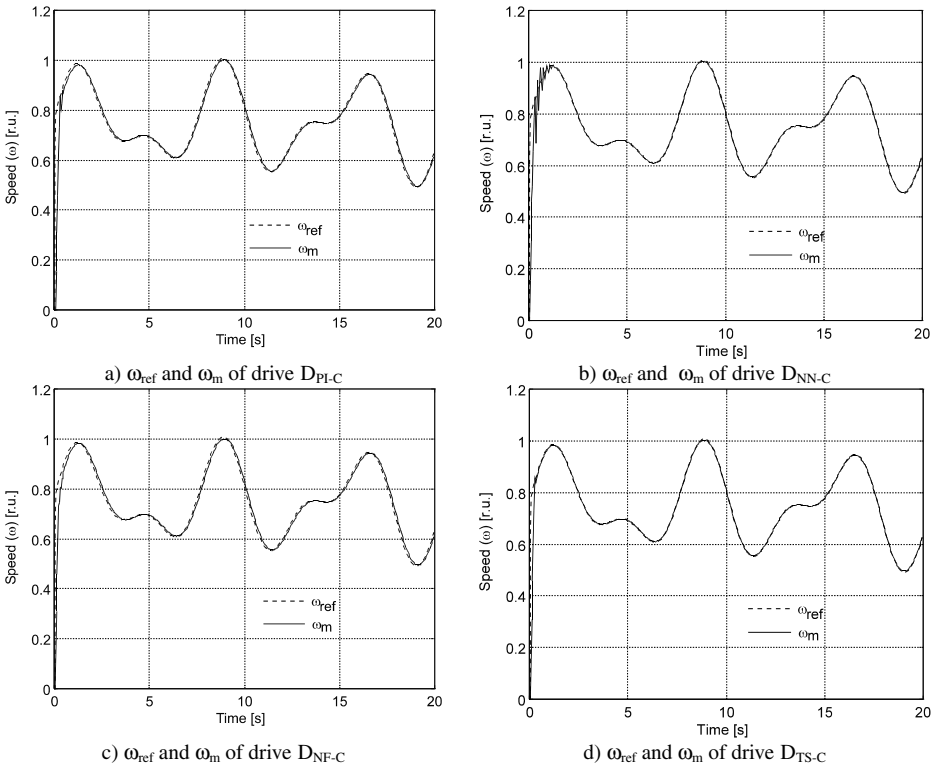

Fig. 3. Reference and actual speed traces with the four types of the speed controller in operating regime 11.

the performance index is indicated in bold for each of the 12 regimes.

An illustration of the simulation results is given in Fig. 3, where the reference and the actual speed are shown for each of the four considered speed controller types. The results apply to the regime number 11.

It follows from Table 7 that the TS based speed controller exhibits the best behaviour in majority of cases. The ANFIS based speed controller is somewhat inferior. Additionally, its structure is significantly more complex. The FNN based speed controller is the worst and it is even inferior to the optimised PI controller.

6. Conclusion

The paper analyses performance of a speed-sensorless DC motor drive in which three different types of the AI speed controllers are used. Speed controllers are designed using the FNN, neuro-fuzzy network, and the self-organising TS rule-based model. Design of the additional components of the control system, such as the speed estimator, the speed predictor and the load torque estimator, is described as well. The outputs of these components that are based on the FNNs serve as the inputs into the speed controller.

For the sake of a better insight into the achievable performance with the AI based speed controllers, a control system based on conventional PI speed controller is designed as well. Simulation of all the drive structures for a number of operating regimes leads to the following conclusions:

- The self-organising TS fuzzy controller appears to be the best solution since it: (1) exhibits the best accuracy in the majority of operating regimes; (2) requires the shortest training time; (3) is characterised with the simple final structure that is convenient for real-time implementation; (4) leads to fuzzy rules that contain simple analytical correlation between the state variables and the control variable.
- The neuro-fuzzy network (ANFIS) is characterised with a rather good accuracy. However, the structure is complex and the long training period is required.
- The FNN is characterised with a shorter training period and a simpler configuration, when compared to the ANFIS. However, the achieved accuracy is inferior to the one obtained with the other two analysed types of the AI based speed controller.
- The reference, optimised conventional PI controller is characterised with the simple design procedure and the simple structure. The accuracy shown in the testing is very good. For two of the twelve operating regimes the best performance was obtained with the optimised PI controller. However, it should be noted that the PI controller parameters were individually optimised for each of the twelve considered operating regimes. The optimised PI controller therefore corresponds to some extent to an ‘adaptive’ PI controller. The performance of such an ‘adaptive’ PI controller is still inferior to the

performance obtained with the both ANFIS and especially self-organising TS based controller (the best performance in three and seven regimes, respectively). Realisation of such an 'adaptive' PI controller would represent a computationally complex solution that would eliminate the major advantage of a PI controller, its simplicity. Needless to say, had a constant parameter PI controller been considered, with its parameters obtained by optimisation for a single operating regime, the results of the PI control for the remaining eleven operating regimes would have been significantly worse than those listed in Table 7.

The extra-complexity of the AI based speed controllers, when compared to the constant parameter PI controller is believed to be an acceptable trade-off between the performance and the realisation requirements. Trained AI based controllers of minimal configuration (i.e. the minimum additional complexity) are considered and substantial improvements in the accuracy are obtained, especially with the self-organising TS rule based model. In addition, the AI based controllers are characterised with very good robustness properties and enable parallelism in implementation.

The results of the study fully justify application of the AI based control techniques in motor drives. Although the study described here is related to a separately excited DC motor drive, the results are directly applicable to so-called vector controlled AC motor drives. This generalisation is possible since the application of vector control principles in AC motor drives in essence converts an AC machine into its DC machine equivalent. Thus, from the point of view of the speed controller, it becomes irrelevant whether the electric drive under consideration utilises an AC or a DC motor.

The main conclusions of this research are believed to be directly applicable in the control of a wider class of industrial processes, where the main requirement is high accuracy. Detailed expert knowledge about the process is not required. The ANFIS approach and the self-organising TS fuzzy model based approach enable grouping of samples in clusters, which may represent individual operating regimes. Combination of such individual regimes, represented by appropriate control rules, enables an adequate description of the complex dynamics of the system under consideration.

Appendix A

Motor model: By using the following base values (index b):

$$U_{ab} = U_{an}, \qquad R_{ab} = U_{ab}/I_{ab}, \qquad I_{ab} = I_{an},$$

$$\psi_b = U_{ab}/\omega_b, \qquad \omega_b = \omega_n, \qquad T_b = \psi_b I_{ab},$$

where index n denotes rated values, motor model can be given in per unit system as (constant rated excitation flux is assumed):

$$\frac{d}{dt}\begin{bmatrix} \omega \\ i_a \end{bmatrix} = \begin{bmatrix} -0.0667 & 1.9393 \\ -4227.1346 & -225.9036 \end{bmatrix}\begin{bmatrix} \omega \\ i_a \end{bmatrix} + \begin{bmatrix} -2.0413 & 0 \\ 0 & 4449.6154 \end{bmatrix}\begin{bmatrix} T_m \\ u_a \end{bmatrix}, \tag{A1}$$

where ω is the angular speed, i_a the armature current, u_a the armature voltage, T_m the load torque.

Simulation model includes armature voltage limiting (120% of the rated value). Power supply is modelled as a first order delay with time constant of 3 ms. Speed is measured with a tacho-generator, whose time constant is 48 ms. Sampling time is 10 ms.

Motor data:

$U_{an} = 260$ V	$R_a = 0.75\ \Omega$	$I_{an} = 1.76$ A
$L_a = 3.32$ mH	$n_n = 3370$ rpm	$\psi_{fn} = 0.7$ Wb
$\omega_n = 352.9$ rad/s	$J = 0.018$ kgm^2	$P_n = 3.9$ kW
$I_{fn} = 0.56$ A	$K_\omega = 0.0012$ Nm/(rad/s)	

References

[1] Haykin S. Neural networks. New York: Macmillan, 1994.
[2] Klir GJ, Yuan B. Fuzzy sets and fuzzy systems—theory and applications. Englewood Cliffs, NJ: Prentice Hall, 1995.
[3] Jang JR, Sun C, Mizutani E. Neuro-fuzzy and soft computing. Englewood Cliffs, NJ: Prentice Hall, 1997.
[4] Narendra SK, Parthasarathy K. Identification and control of dynamical systems using neural networks. IEEE Trans Neural Networks 1990;1(1):4–27.
[5] Narendra SK. Neural networks for control: theory and practice. Proc IEEE 1996;84(10):1385–406.
[6] Gupta MM, Rao DH. Neuro-control systems: theory and application. Piscataway, NJ: IEEE Press, 1994.
[7] Hoskins CJ, Kaliyuar KM, Himmemblau DM. Fault diagnosis in complex chemical plants using artificial neural networks. AIChE J 1991;37:137–41.
[8] Peng TM, Hubele NF, Karady GG. Advancement in the application of neural networks for short-term load forecasting. IEEE Trans Power Systems 1992;7(1):250–7.
[9] Ku CC, Lee KY. Diagonal recurrent neural networks for dynamic systems control. IEEE Trans Neural Networks 1995;6(1):144–55.
[10] Qin SZ, Su HT, McAvoy TJ. Comparison of four neural net learning methods for dynamic system identification. IEEE Trans Neural Networks 1992;3(1):122–30.
[11] Li YF, Lau CC. Development of fuzzy algorithms for servo systems. IEEE Control Systems Mag 1989;9(3):65–72.
[12] Hang CC, Ho WK, Lee TH. Knowledge based PID control: heuristics and implementation. In: Gupta MM, Sinha NK, editors. Intelligent control systems—theory and applications, Piscataway, NJ: IEEE Press, 1996. p. 345–82.
[13] Li HX, Gatland HB. A new methodology for designing a fuzzy logic controller. IEEE Trans Systems, Man, Cybernet 1996;25(3): 505–12.
[14] Li W. A method for design of a hybrid neuro-fuzzy control system based on behaviour modelling. IEEE Trans Fuzzy Systems 1997;5(1): 128–37.

[15] Galithet S, Foulloy L. Fuzzy controllers: synthesis and equivalencies. IEEE Trans Fuzzy Systems 1995;3(2):140–7.
[16] Kukolj D, Kuzmanovic S, Levi E, Kulic F. Design of a near-optimal, wide-range fuzzy logic controller. Fuzzy Sets Systems, accepted.
[17] Klawonn F, Kruse R. Constructing fuzzy controller from data. Fuzzy Sets Systems 1997;85:177–93.
[18] Chen JQ, Xi YG, Zhang ZJ. A clustering algorithm for fuzzy model identification. Fuzzy Sets Systems 1998;98:319–29.
[19] Homaifar A, McCormick E. Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms. IEEE Trans Fuzzy Systems 1995;3(2):129–34.
[20] Isaka S, Sebald AV. An optimisation approach for fuzzy controller design. IEEE Trans Systems, Man, Cybernet 1992;22(6):1469–73.
[21] Juang CF, Lin CT. An on-line self-constructing neural fuzzy inference network and its applications. IEEE Trans Fuzzy Systems 1998; 6(1):12–32.
[22] Jang JSR, Sun CT. Neuro-fuzzy modelling and control. Proc IEEE 1995;83(3):378–406.
[23] Setnes M, Babuška R, Verburger HB. Rule-based modeling: precision and transparency. IEEE Trans Systems, Man, Cybernet—Part C 1998;28(1):165–9.
[24] Levine WS. The control handbook. Boca Raton, FL/New York: CRC/ IEEE Press, 1996.
[25] Orlowska-Kowalska T, Migas P. Neural network estimator for the induction motor drive. Power Electronics and Motion Control Conference Proceedings, Prague, Czech Republic, 1998. p. 89–94.
[26] Krause PC, Wasynczuk O. Electromechanical motion devices. New York: McGraw Hill, 1989.
[27] Nelder JA, Mead R. A Simplex method for function minimisation. Computer J 1965;7:308–13.